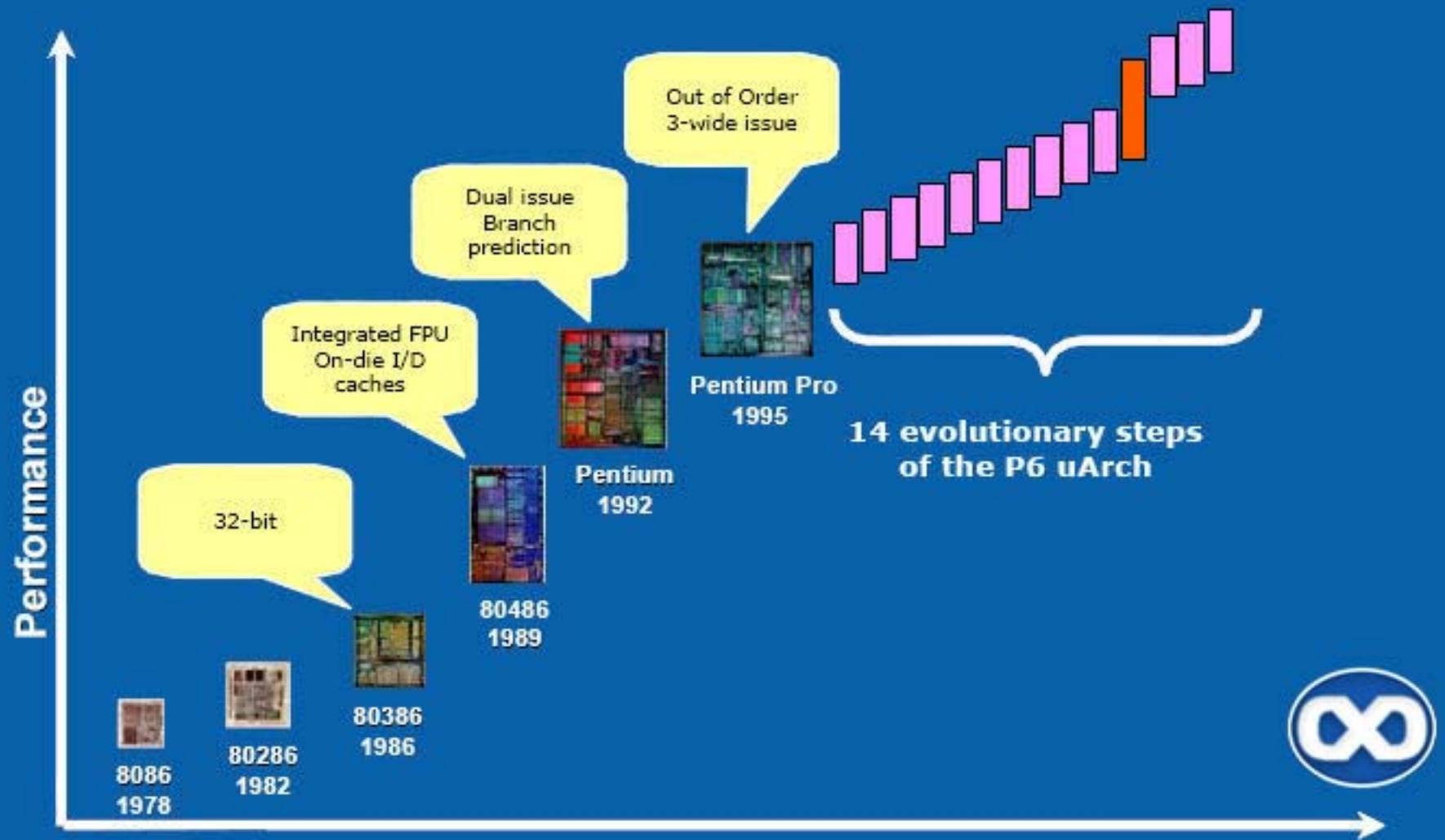
Future CPU Architectures: The Shift From Traditional Models

Doug Carmean

A 30 year Retrospective



CPU profit margins are decreasing

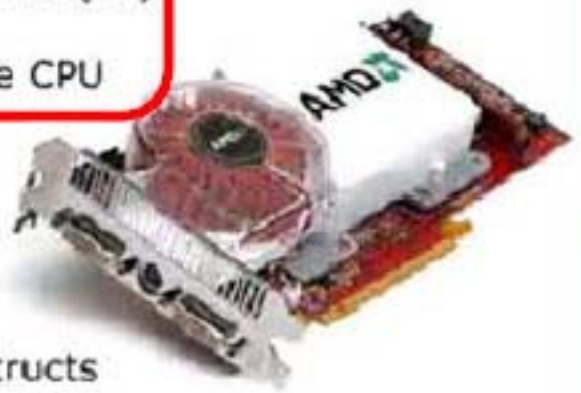
GPU margins are increasing

The industry begins another significant transformation

Stream Computing

AMD
Smarter Choice

- General purpose computation on the GPU (GPGPU)
  - Unleashes the tremendous floating point (FP) performance of the GPU
  - An order of magnitude faster than the CPU
- Changes to GPU architectures has enabled this
  - 32-bit FP
  - Long shader programs
  - Branching and looping program constructs
- Software has remained the largest obstacle to widespread adoption
- Close-To-Metal (CTM) layer is key to unlocking the power of GPGPU



CUDA & GPU Computing

- CUDA is a completely new architecture and programming model for general-purpose computation on GPUs
- Hardware and Software designed together
  - NOT a new driver for old GPU architectures
- Data-parallel computing with thousands of threads
- Parallel Data Cache helps increase arithmetic intensity for massive speedups
- Program in C
- BLAS and FFT libraries

YIKES!

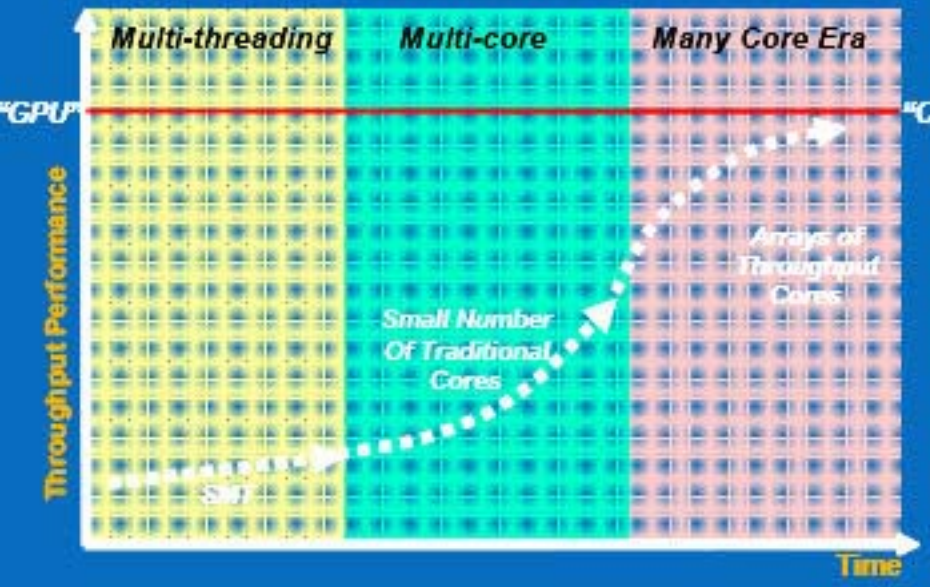
Computing Evolution: A Collision Course

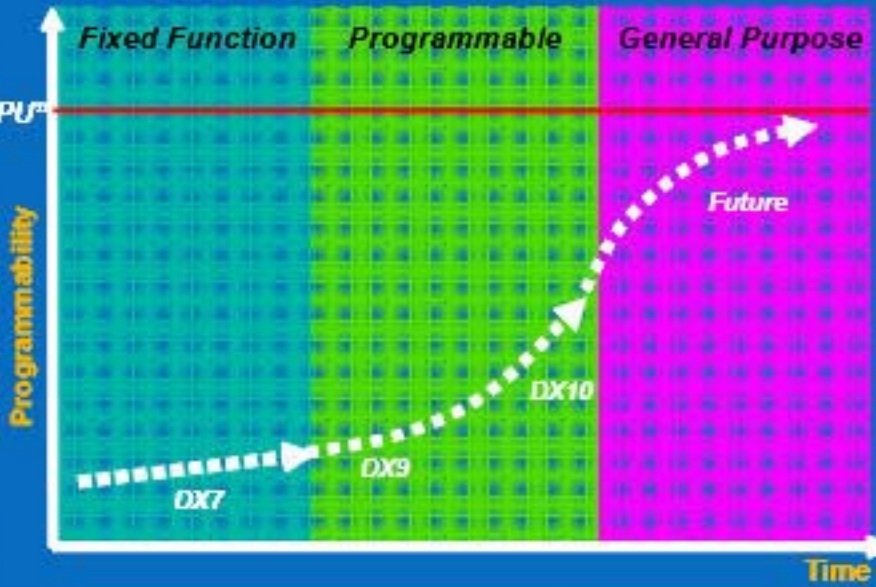
CPU:

- Evolving toward throughput computing
- Motivated by energy-efficient performance

GPU:

- Evolving toward general-purpose computing
- Motivated by higher quality graphics and GP-GPU usages





Battle for control of the computing platform

Optimizing for Power Efficiency

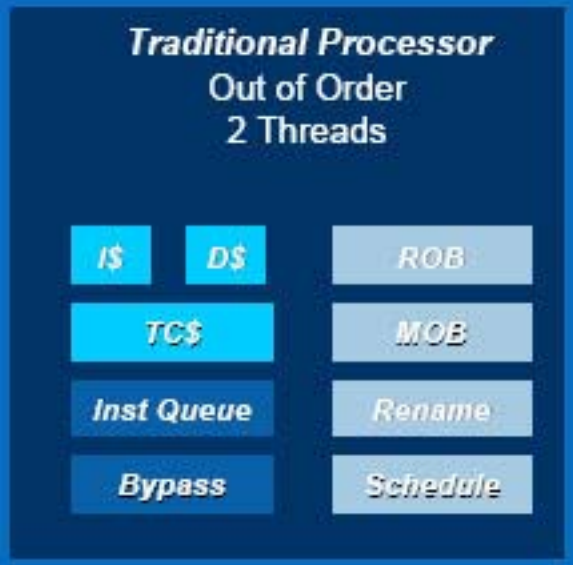
Increase power efficiency with explicit parallelism:

- VLIW & Vectors:
  - Good when there is tight synchronization
  - Bad when data being processed follows different paths
- Threads
  - Bad when there is tight synchronization
  - Fine when data being processed follows different paths

Push each technique to a practical point

- Vector efficiency starts to degrade as vector length increases
- Threading efficiency starts to degrade as communication increases

Consider the comparison

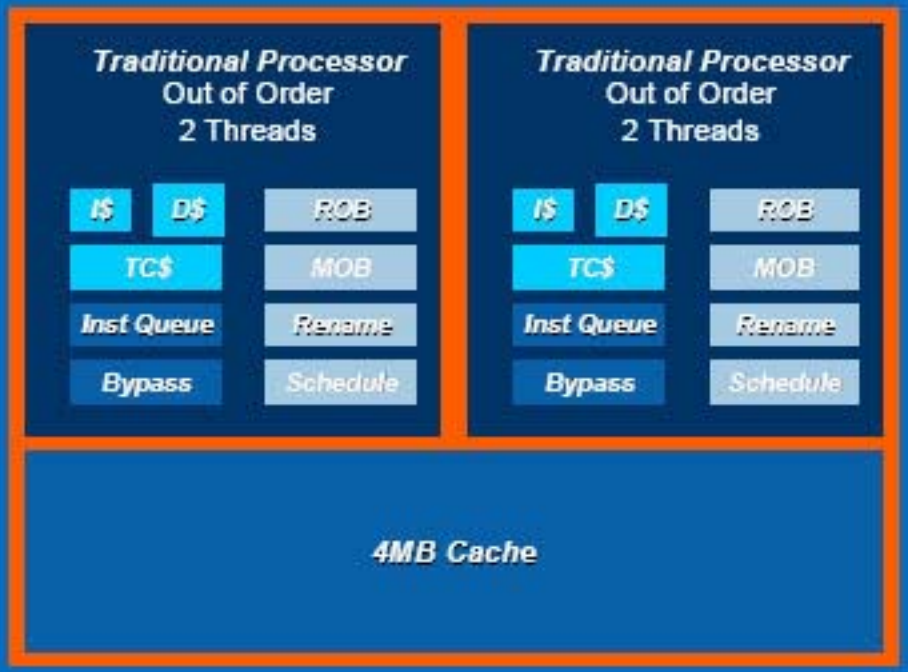


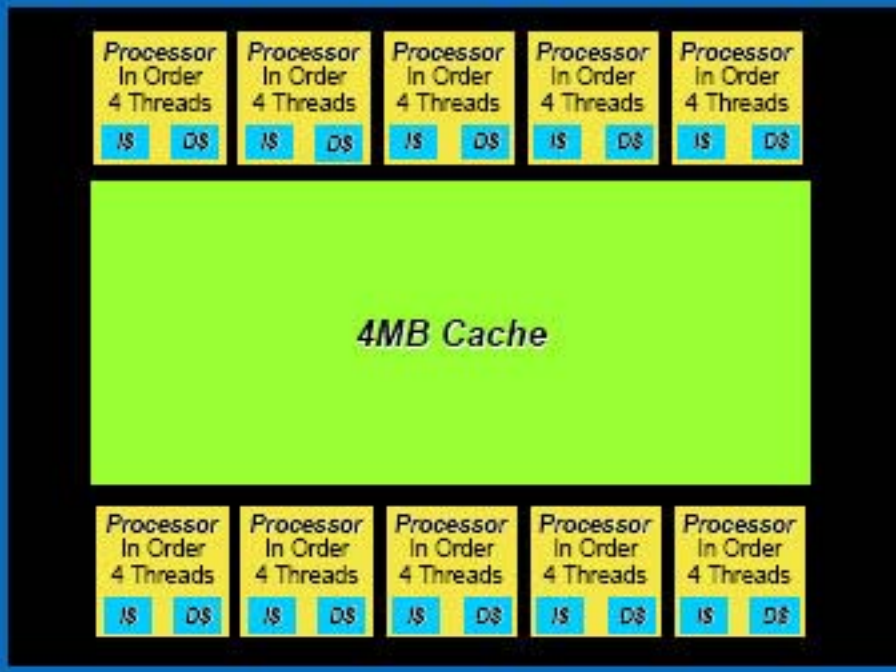
Consider the comparison

	Traditional Core	Throughput Core	
uArch	Out of Order	In Order	
Size	50	10	mm^2
Power	37.5	6.25	W
Freq	4	4	GHz
Threads	2	4	
Single Thread	1	0.3	Relative Performance
Vector	4 (128-bit)	16 (512-bit)	
Peak Throughput	32	128	GFLOPS
Area Capacity	0.6	13	GFLOPS/mm
Power Capacity	0.9	20	GFLOPS/W

Potential for 20x power efficiency improvement

Potential Architecture Choices

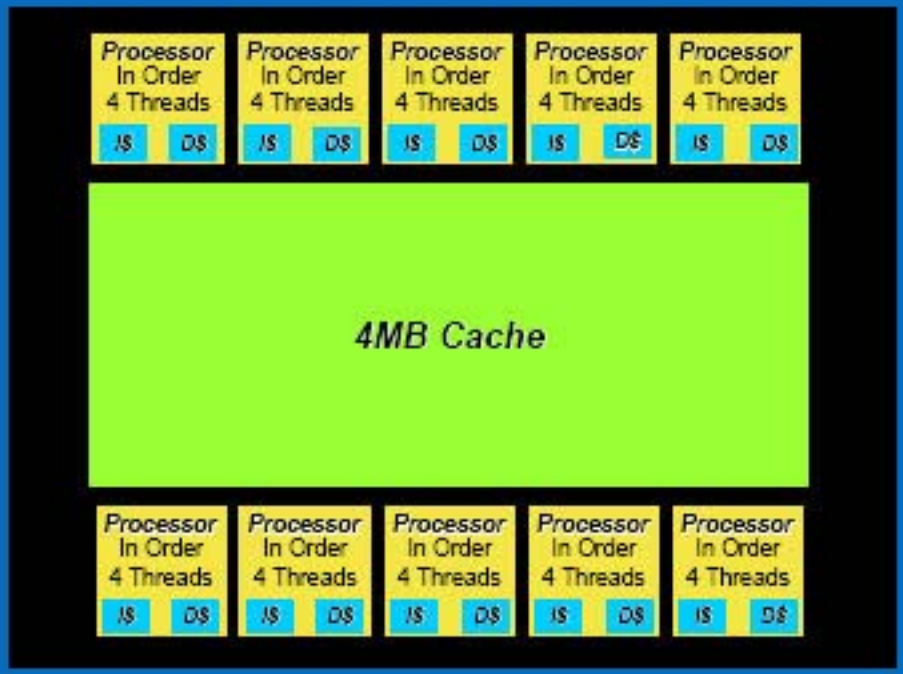




140mm^2
90W
1.0x ST
64 GFLOPS

140mm^2
90W
0.5x ST
1200 GFLOPS

What is this thing?



- Chip Multi Processors
  - >10 cores on a die
  - SMP like scaling
- Threaded Processors
  - 4 threads per core
  - SMT like scaling
- Vector machine
  - 16-wide vector
  - Vector tradeoffs
- VLIW Instructions
  - VLIW tradeoffs

Write good code for that!

We'll provide the following

Caches with decent behavior

- They are coherent
- Relatively low latency (~10 clock L2)

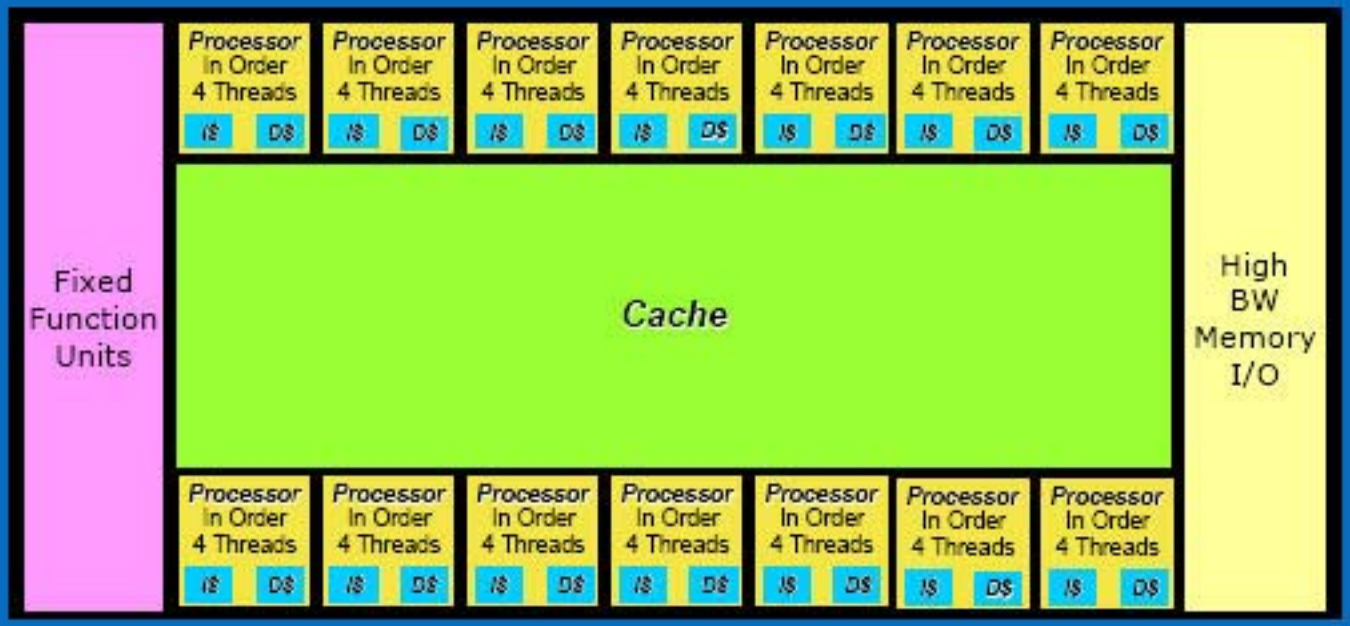
Reasonable communication latency

- <20 clock locks, semaphores
- Primitives for thread synchronization

Good bandwidth

- 1TB/s on die aggregate bandwidth
- >150GB/s off die bandwidth

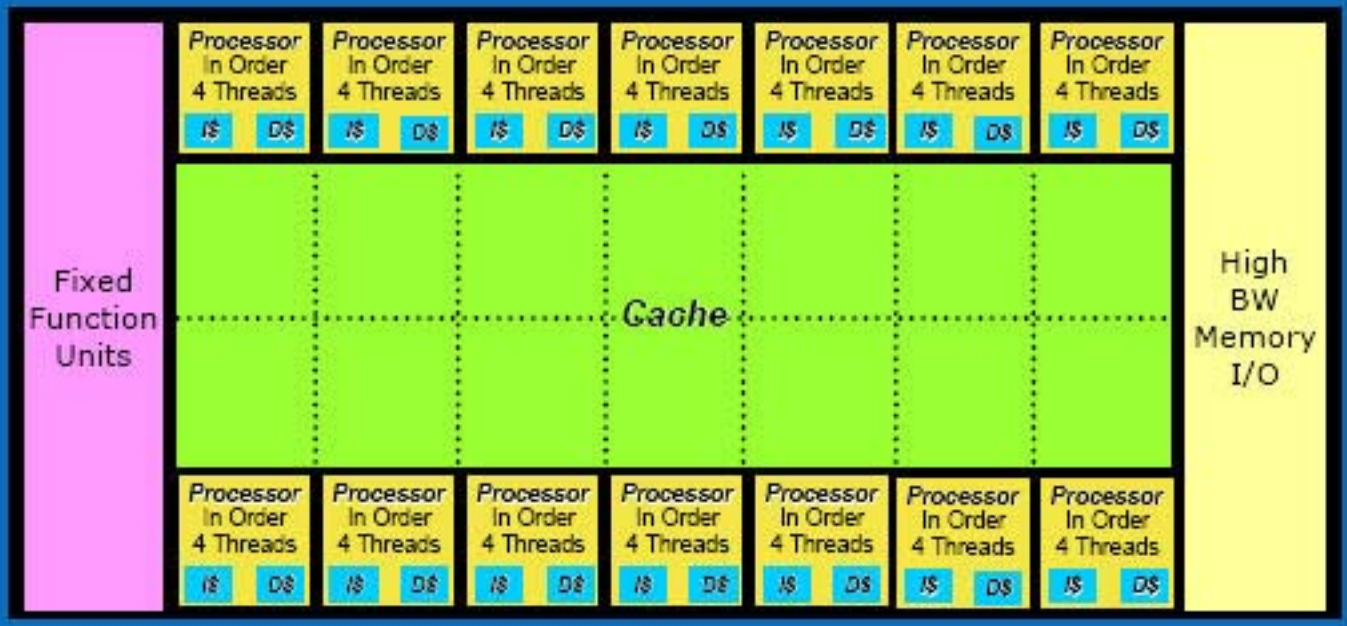
Pulling it together:



Unified Cache

- Data Sharing
- Efficient, in-memory communication
- Dynamic sizing (it is a cache)

And this:



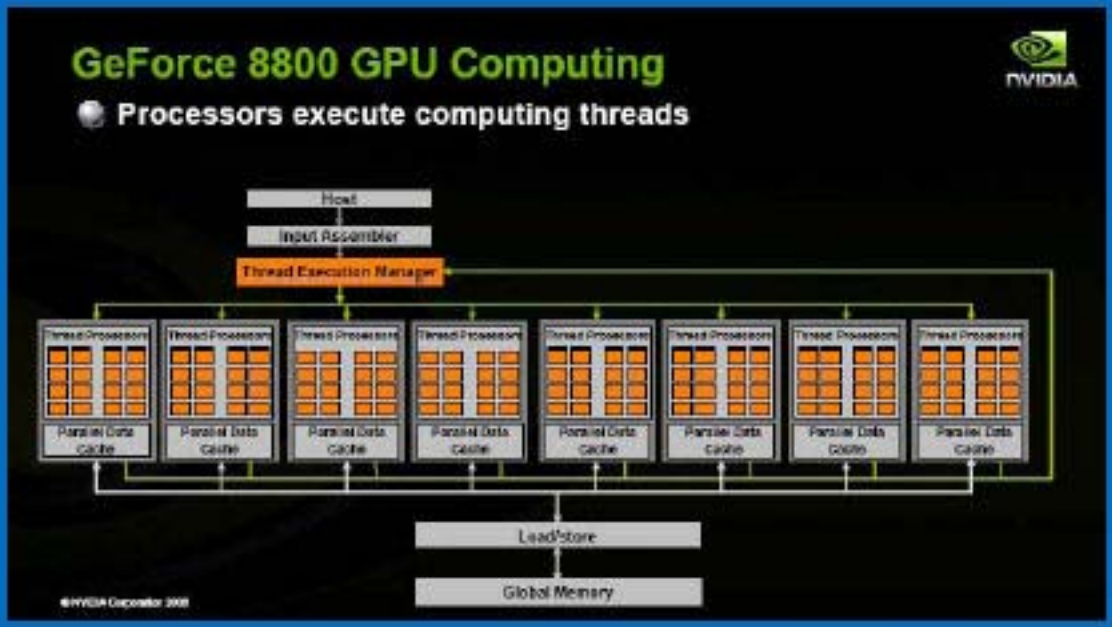
Dynamic cache partitioning

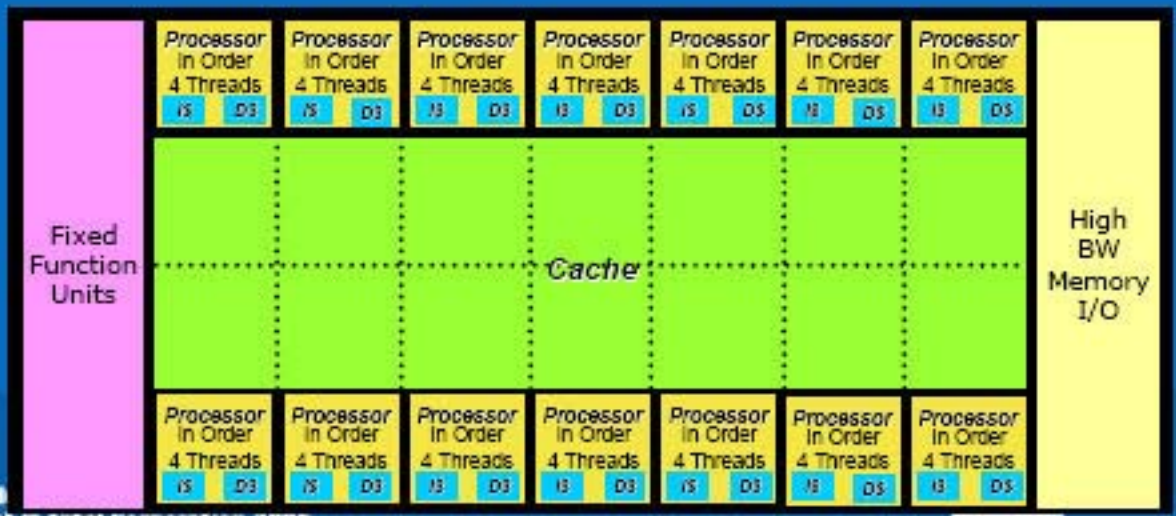
- Private caches for high, aggregate bandwidth
- Arbitrary data replication
- Data swapping between partitions
- Heuristic based configuration changes
- Will likely come with non-uniform access latencies

The War is On!

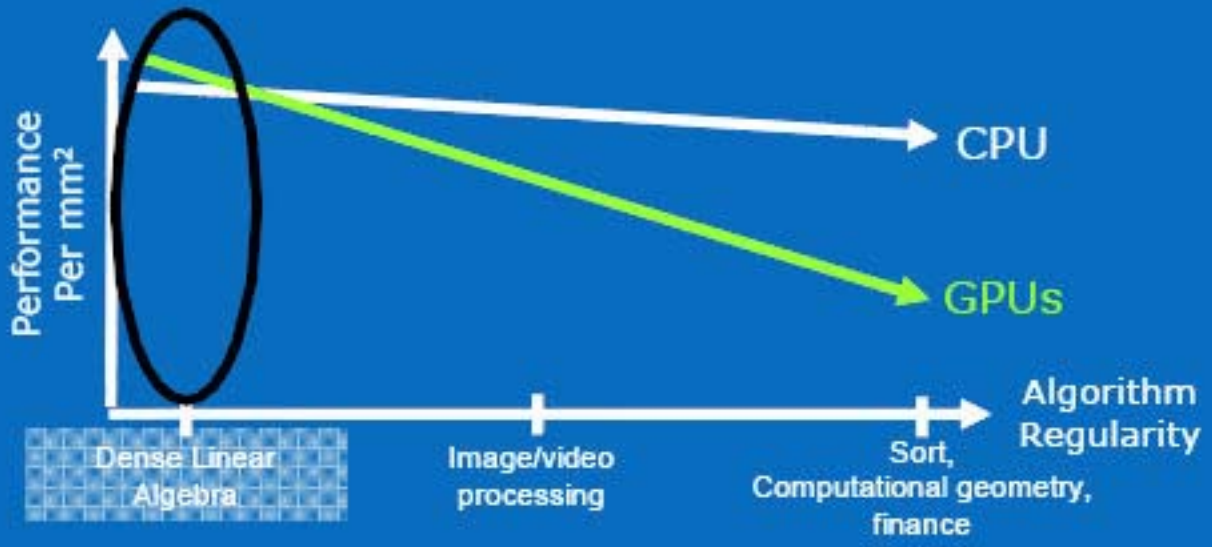
The Competition

See any similarities?



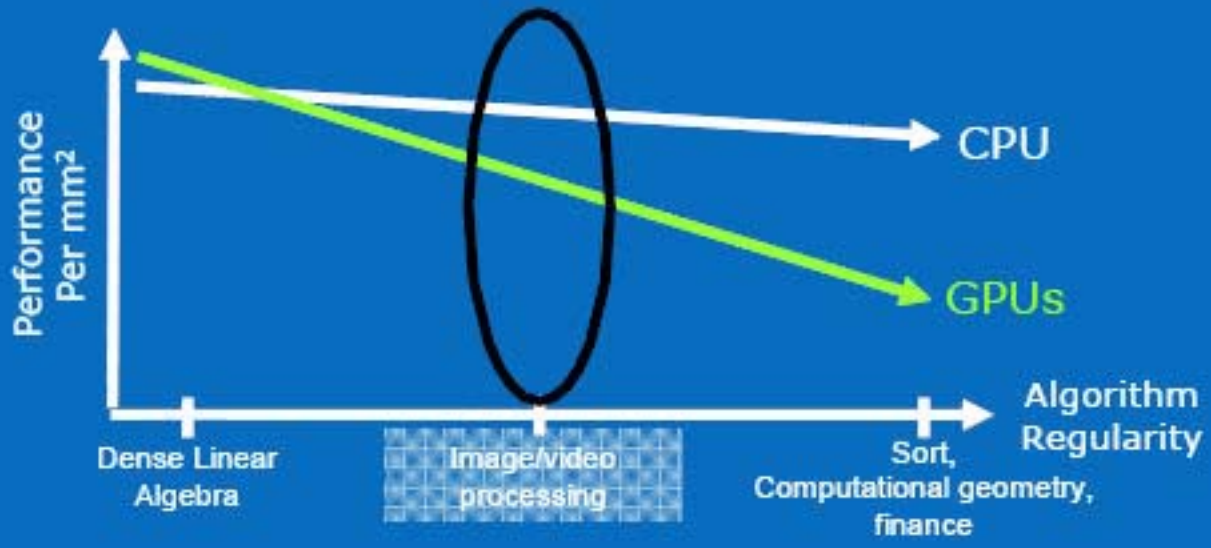


Algorithm Examples



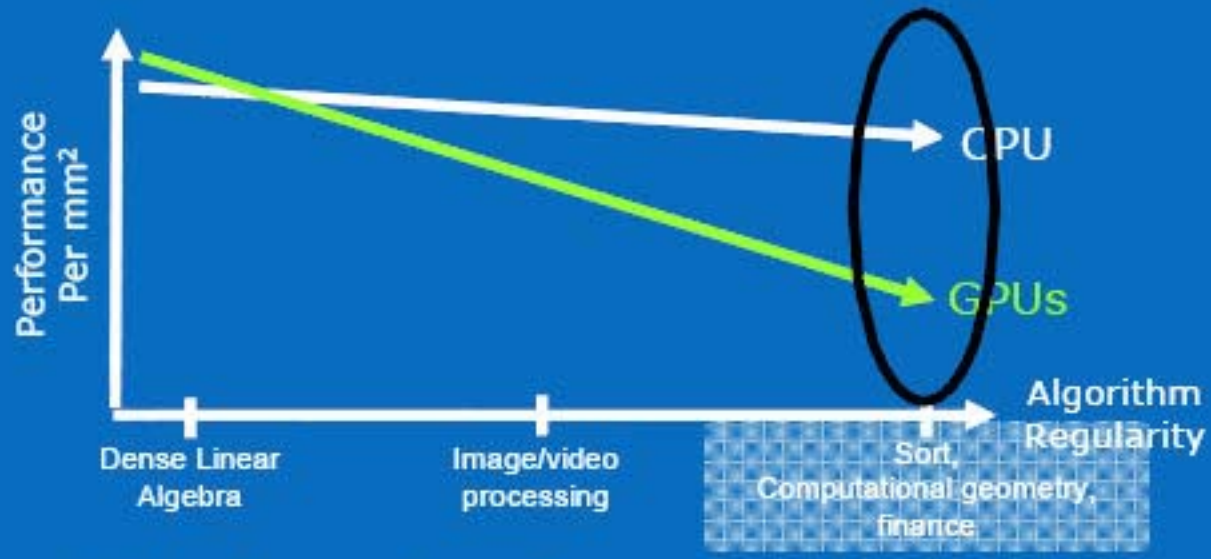
- Dense Linear Algebra
  - No control flow
  - Dense and regular data structures
  - Simple reductions
- GPU Home Field Advantage
  - GPUs high throughput and low overhead maximize compute density
  - In general, GPUs will outperform CPUs on these algorithms

Algorithm Examples



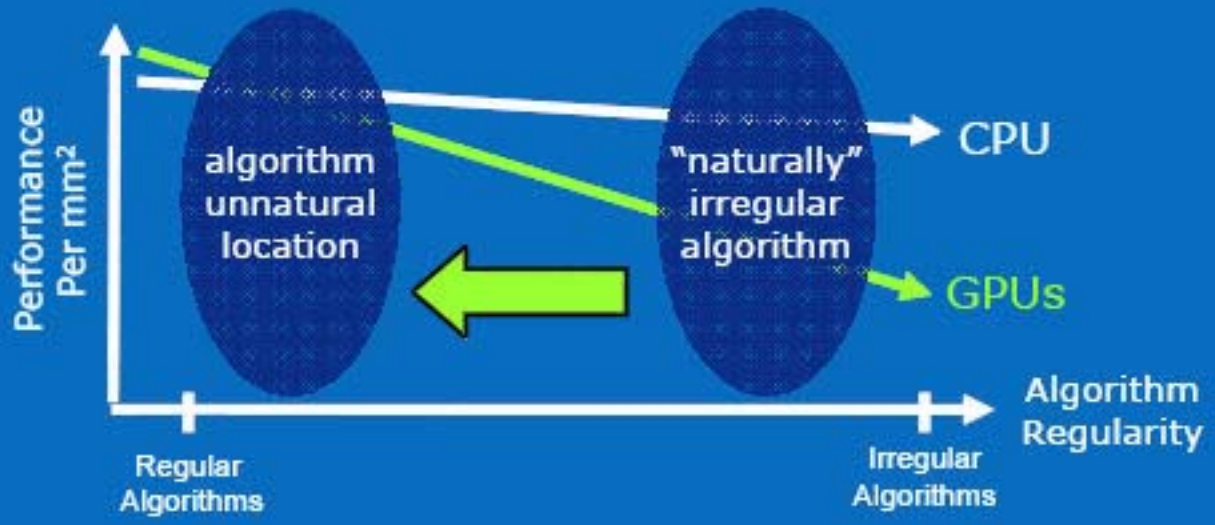
- Image/video processing
  - No control flow
  - Spatially, 2D processing
  - Including the time dimension, 3D processing
  - Localized communication between elements

Algorithm Examples



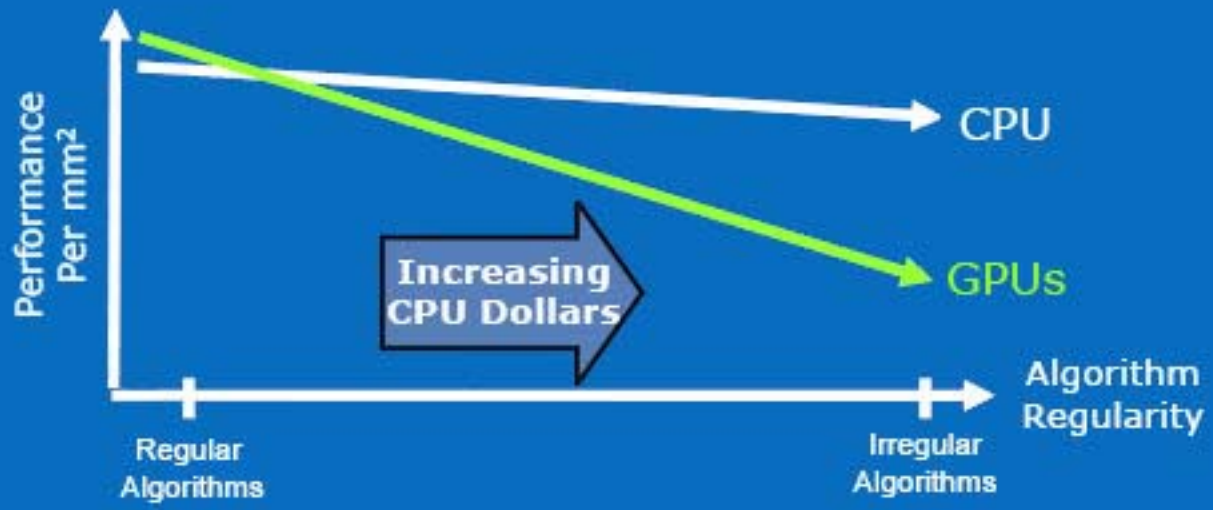
- Sort, computational geometry, finance
  - Modest control flow
  - Sparse/Irregular data structures
  - Irregular communication between elements
- CPU Territory
  - General purpose features vital for software efficiency
  - Latency sensitive applications

Dragging Applications "To the Left"



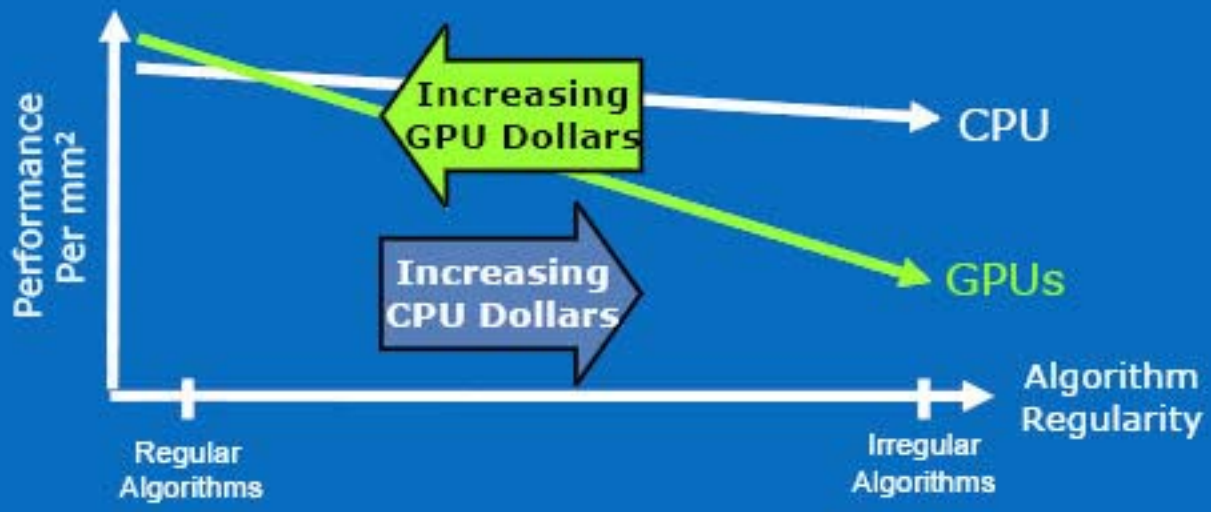
- Applications have a natural location on the continuum
- Possible to reformulate applications to use more regular algorithms
  - Usually requires significant programmer effort
  - Often implies algorithms not as well suited to application
  - HW efficiency improvements more than offset SW inefficiency
- Optimal solution is function of overall efficiency (HW, SW, programmer)

Making Money



- CPU vendors want to maintain relevancy of Irregular Algorithms
  - Most relevant applications are naturally Irregular
  - Currently, GPUs only run subset of applications using Regular Algorithms
- Drive to increase the need for programmable hardware
  - Improve throughput performance
  - Robust tools and infrastructure

Making Money



- GPU vendors are trying to accelerate this reformulation
  - GPGPU (www.gpgpu.org)
  - Physics on GPUs (Havok, ATI)
  - GPU Gems
  - Cg, Brook
  - ATI's DPVM, Peakstream
- Also improving the programmability of the GPU hardware

There is a Battle Waging in Your PC

CPUs and GPUs are battling for control

- CPUs are dramatically improving throughput performance
- GPUs are dramatically improving programmability

Next Generation GPUs will look a lot like CPUs

- Nvidia's G80, ATI's R600
- Or, is it that CPUs will look a lot like GPUs?

The war is being fought over applications

- GPUs are dragging applications "To the Left"
- CPUs are attempting to hold applications on the Right